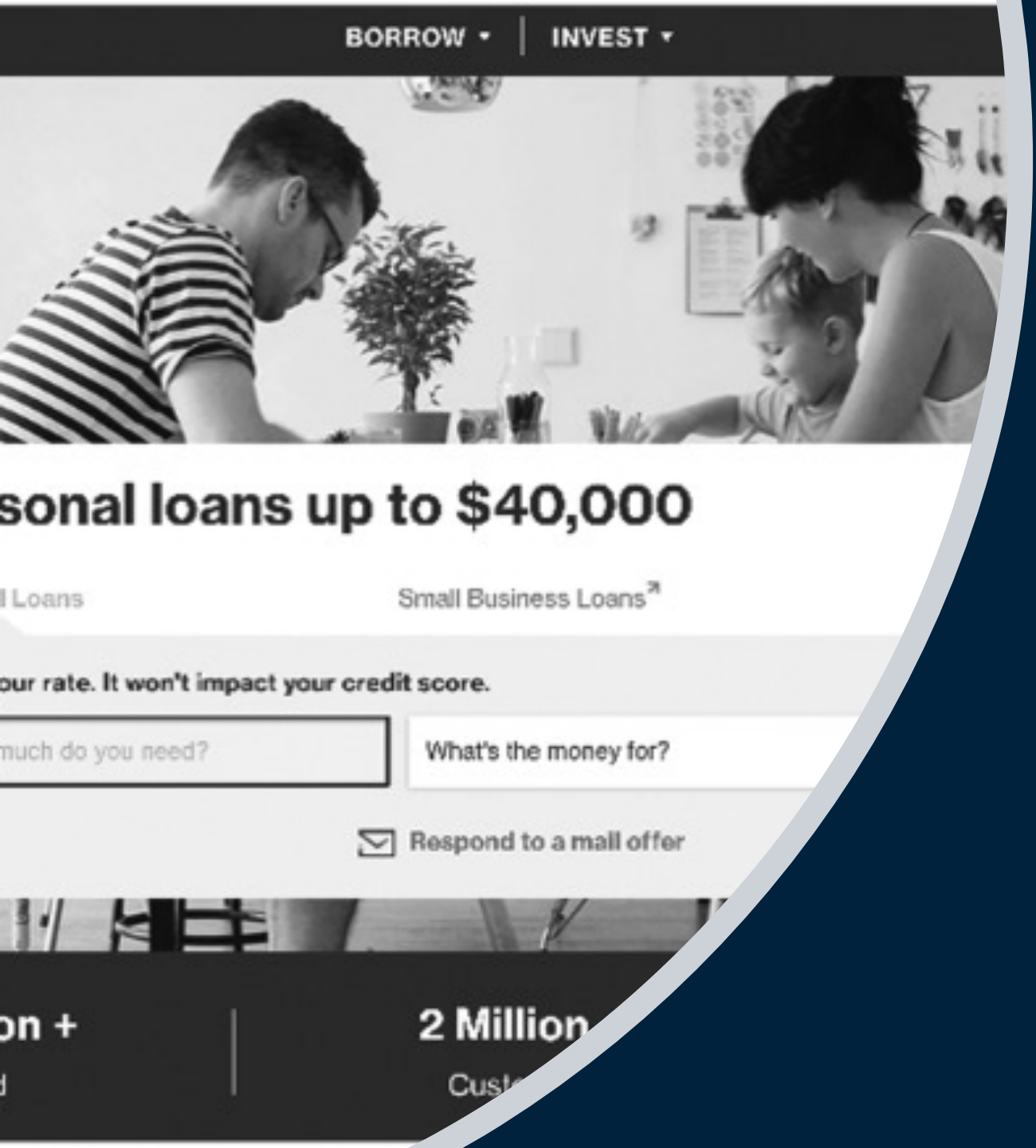
LendingClub

Data Science in the service of Financial Democratization

Lecture plan

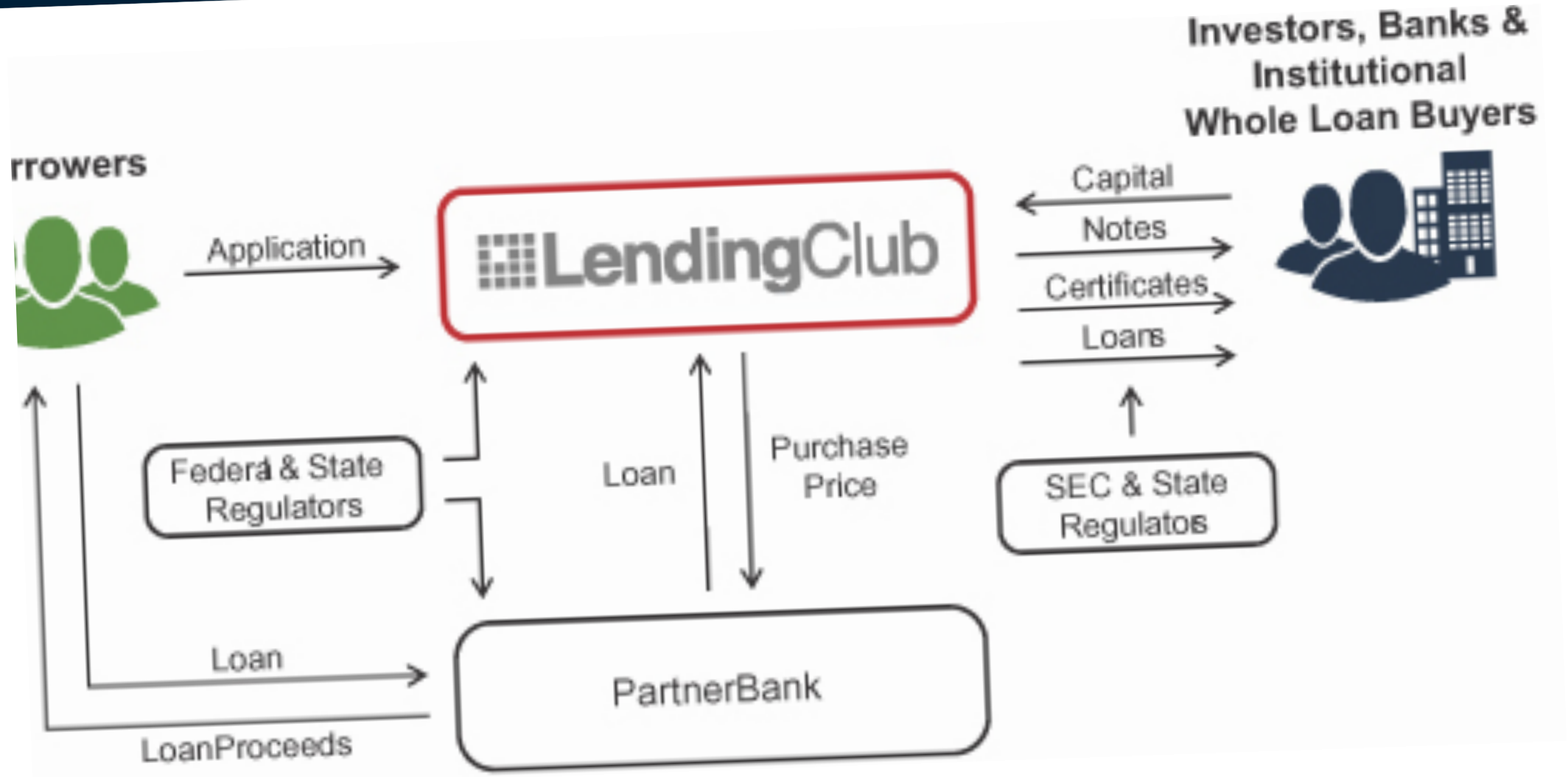
- Vectorization in NumPy
- Lending Club
  - What is Lending Club
  - Mechanics of Lending Club
- Lending Club Data
  - Features in the data set
  - Repayment behavior
  - Estimating performance
- Cleaning the data



Lending Club

- Peer to peer lending as a bank alternative


Pay
LendingClub
LendingClub screens borrowers, facilitates the transaction, and services the loans
Transfer
$100
Borrowers
earn savings and financial control
Investors
earn solid returns
Founded in 2007 and
headquartered in San Francisco, CA


rrowers
Application
LendingClub
Investors, Banks &
Institutional
Whole Loan Buyers
Capital
Notes
Certificates
Loans
Federal & State
Regulators
Loan
Purchase
Price
SEC & State
Regulators
Loan
PartnerBank
LoanProceeds

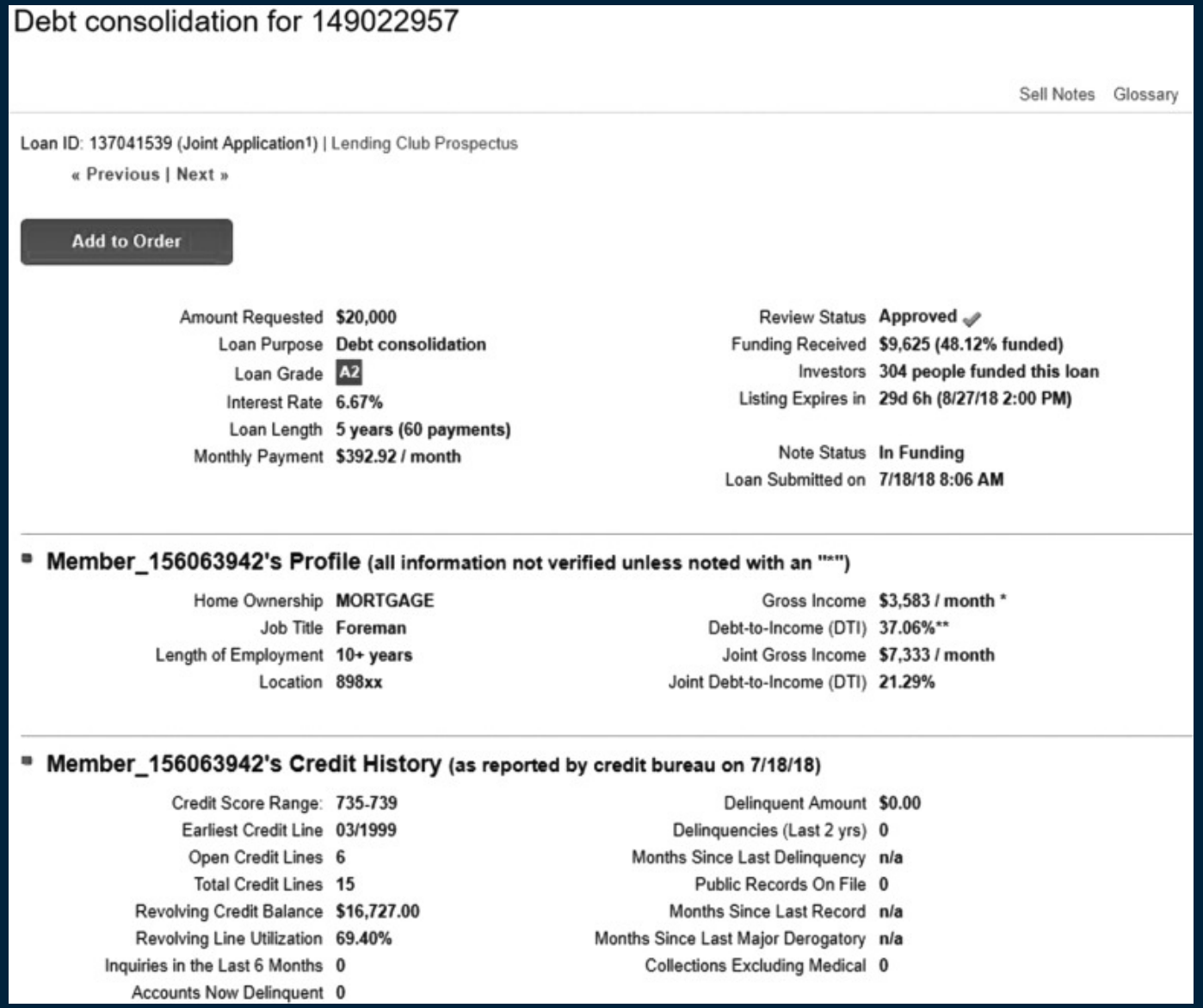
Lending Club UI

- A loan is broken down into notes
- Investors purchase individual notes
- A loan is not issued before full funding is guaranteed

Investment	Rate	Term	FICO®	Amount	Purpose	% Funded	Amount / Time Left
$0	B 5 11.99%	36	670-674	$7,000	Credit Card Payoff	28%	$5,025 29 days
$0	C 3 14.08%	36	685-689	$35,000	Loan Refinancing & Consolidation	41%	$20,375 29 days
$0	C 3 14.08%	36	670-674	$8,000	Credit Card Payoff	15%	$6,750 29 days
$0	C 1 12.62%	60	660-664	$20,000	Other	93%	$1,275 25 days
$0	C 1 12.62%	60	735-739	$34,700	Credit Card Payoff	68%	$10,900 27 days
$0	C 2 13.59%	60	715-719	$22,500	Loan Refinancing & Consolidation	73%	$5,925 27 days
$0	B 5 11.99%	60	715-719	$28,000	Other	84%	$4,375 28 days
$0	D 1 17.09%	36	715-719	$30,000	Loan Refinancing & Consolidation	95%	$1,325 28 days
$0	D 2 18.06%	36	700-704	$10,000	Other	62%	$3,775 28 days
$0	D 3 19.03%	36	675-679	$6,000	Other	85%	$875 28 days

Data available on each loan application

- **Loan request information**
- **Applicant information**
- **Credit history**



Debt consolidation for 149022957

Sell Notes Glossary

Loan ID: 137041539 (Joint Application1) | Lending Club Prospectus

« Previous | Next »

Add to Order

Amount Requested	$20,000	Review Status	Approved
Loan Purpose	Debt consolidation	Funding Received	$9,625 (48.12% funded)
Loan Grade	A2	Investors	304 people funded this loan
Interest Rate	6.67%	Listing Expires in	29d 6h (8/27/18 2:00 PM)
Loan Length	5 years (60 payments)		
Monthly Payment	$392.92 / month	Note Status	In Funding
		Loan Submitted on	7/18/18 8:06 AM

Member_156063942's Profile (all information not verified unless noted with an "*")

Home Ownership	MORTGAGE	Gross Income	$3,583 / month *
Job Title	Foreman	Debt-to-Income (DTI)	37.06%**
Length of Employment	10+ years	Joint Gross Income	$7,333 / month
Location	898xx	Joint Debt-to-Income (DTI)	21.29%

Member_156063942's Credit History (as reported by credit bureau on 7/18/18)

Credit Score Range:	735-739	Delinquent Amount	$0.00
Earliest Credit Line	03/1999	Delinquencies (Last 2 yrs)	0
Open Credit Lines	6	Months Since Last Delinquency	n/a
Total Credit Lines	15	Public Records On File	0
Revolving Credit Balance	$16,727.00	Months Since Last Record	n/a
Revolving Line Utilization	69.40%	Months Since Last Major Derogatory	n/a
Inquiries in the Last 6 Months	0	Collections Excluding Medical	0
Accounts Now Delinquent	0		

What is the Investor’s basic decisions?

- In which loan to invest?
- How much to invest?

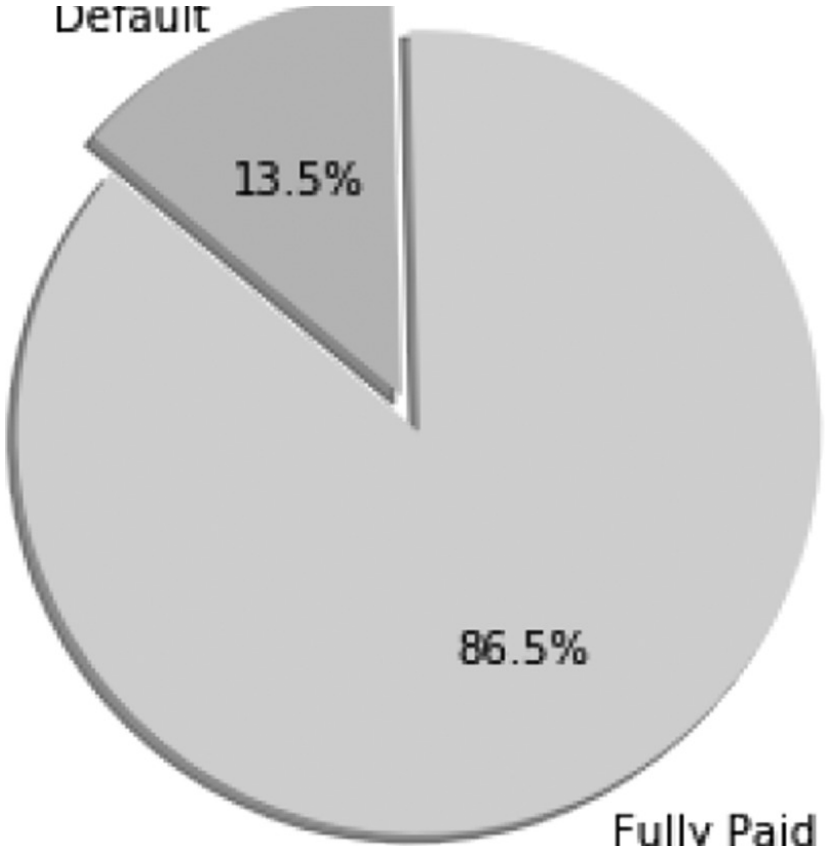
What are the risks the investor is exposed to?

- loaner defaults on part or all of the loan
- The loaner will be late returning the loan
- The loaner will return the loan ahead of time
  - Why is this a risk?
    - Because the loaner pays interest only on the time the loan is out

What is the investor’s objective making these decisions?

- Maximize return
  - The lender can trade off the frequency of defaults with the interest they are receiving

Loan Grades



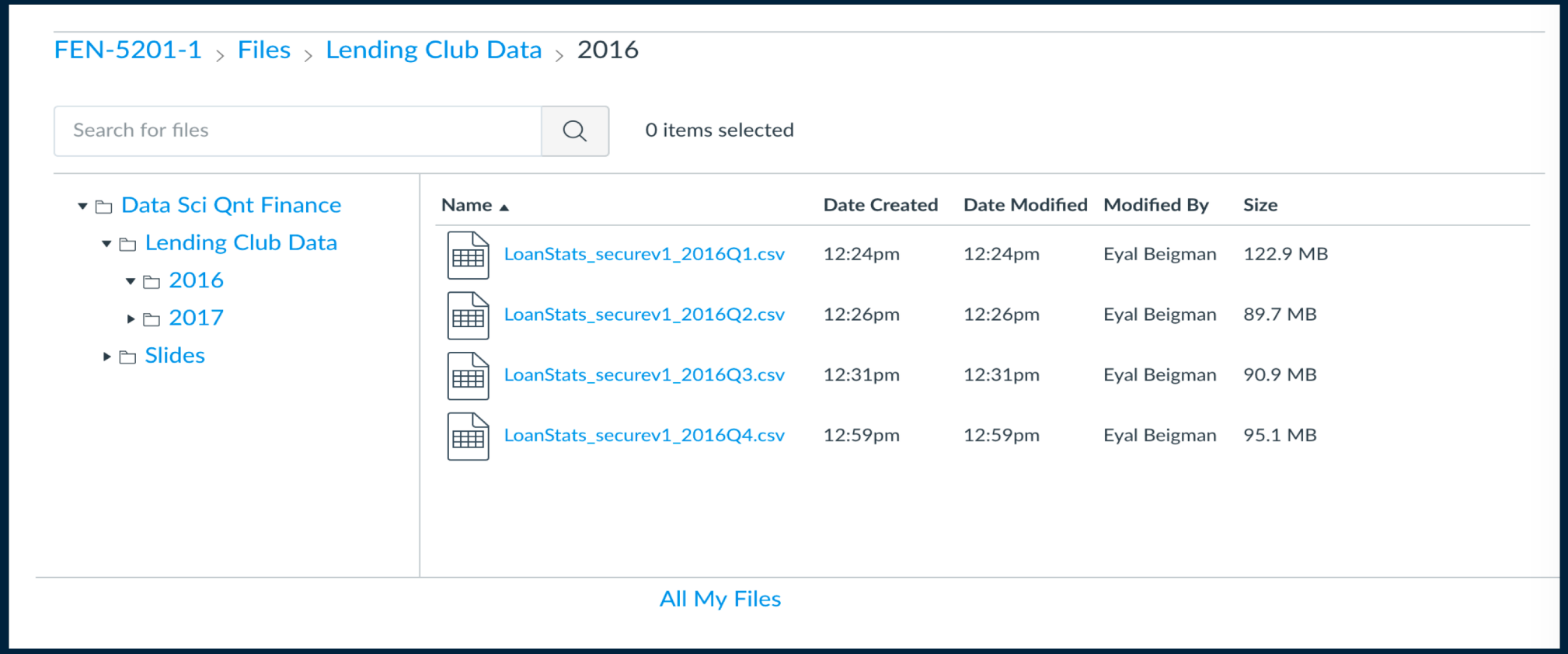
				Mean return			
Grade	*% of loans*	*% Default*	*Av. interest*	*M1*	*M2*	*M3 (1.2%)*	*M3 (3%)*
A	16.68	6.33	7.22	1.66	3.89	2.05	3.71
B	28.86	13.48	10.85	1.58	5.01	2.02	3.68
C	27.99	22.41	14.07	0.62	5.39	1.39	3.02
D	15.44	30.37	17.54	0.05	5.71	0.92	2.51
E	7.59	38.83	20.73	−0.91	5.95	0.11	1.64
F	2.72	44.99	24.47	−1.43	6.43	−0.44	1.05
G	0.73	48.16	27.12	−2.58	6.66	−1.40	0.05

What is the investor’s objective making these decisions?

- Maximize return
  - The lender can trade off the frequency of defaults with the interest they are receiving

Lending Club Data

Where is it?



id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	verification_status	issue_d	loan_status
138894113		12000	12000	12000	36 months	8.46%	378.59	A	A5	janitor	3 years	RENT	36500	Verified	Sep-18	Current
141060314		24000	24000	24000	36 months	8.46%	757.18	A	A5	RN	10+ years	MORTGAGE	115000	Source Verified	Sep-18	Current
141103661		32000	32000	32000	60 months	20.89%	863.73	D	D4	Licensed refrigeration technician	5 years	RENT	65000	Source Verified	Sep-18	Current
141042272		12000	12000	12000	36 months	6.11%	365.67	A	A1	PACS Imaging Tech.	10+ years	MORTGAGE	64000	Source Verified	Sep-18	Current
140902382		12000	12000	12000	60 months	22.35%	333.82	D	D5		n/a	MORTGAGE	38400	Source Verified	Sep-18	Current
141089205		4000	4000	4000	36 months	7.84%	125.06	A	A4	Manager	3 years	RENT	45000	Not Verified	Sep-18	Current
140892631		10650	10650	10650	36 months	7.84%	332.95	A	A4		n/a	RENT	28000	Verified	Sep-18	Current
141092636		9000	9000	9000	36 months	6.11%	274.25	A	A1		n/a	OWN	25000	Not Verified	Sep-18	Current

Lending Club data - Table Format

Ingesting Data Into Pandas

In [73]:
```
def ingest_files(directory):
    '''
    Function ingests files in specified directory into a pandas dataframe. It will return a
    dictionary containing these dataframes, keyed by the file name. We assume the directory
    is a replication of 'Lending Club Data->2016' directory on Canvas.
    '''
    # If the directory has no trailing slash, add one
    if directory[-1] != "/":
        directory = directory + "/"
    all_files = os.listdir(directory)
    output = {}

    print("Directory " + directory + " has " + str(len(all_files)) + " files:")
    for i in all_files:
        print("    Reading file " + i)
        output[i] = pd.read_csv(directory + i, dtype = str, skiprows = 1)
        # Remove lines with non-integer IDs
        invalid_rows = (output[i].id.apply( lambda x : is_integer(x) == False ))
        if invalid_rows.sum() > 0:
            print("        Found " + str(invalid_rows.sum()) + " invalid rows which were removed")
            output[i] = output[i][invalid_rows == False]
    return output

files_2016 = ingest_files(dir_2016)
data_2016 = pd.concat(files_2016.values()).reset_index(drop = True)
```

Choosing desired columns

Create new data frame including only columns of interest

In [5]:
```
# Identify the columns we'll be keeping from the dataset
cols_to_pick = ['id','loan_amnt','funded_amnt','term','int_rate',
                'installment','grade','emp_length', 'home_ownership',
                'annual_inc','verification_status','issue_d',
                'loan_status','purpose','dti', 'delinq_2yrs',
                'earliest_cr_line','open_acc','pub_rec', 'fico_range_high',
                'fico_range_low', 'revol_bal','revol_util', 'total_pymnt',
                                                'last_pymnt_d', 'recoveries']
final_data = data_2016[cols_to_pick].copy()
print("Starting with " + str(len(final_data)) + " rows")
```

```
Starting with 434407 rows
```

Typecast

In [12]:
```
# Identify the type of each of these column
float_cols = ['loan_amnt', 'funded_amnt', 'installment', 'annual_inc',
                  'dti', 'revol_bal', 'delinq_2yrs', 'open_acc', 'pub_rec',
                          'fico_range_high', 'fico_range_low', 'total_pymnt', 'recoveries']
cat_cols = ['term', 'grade', 'emp_length', 'home_ownership',
                 'verification_status', 'loan_status', 'purpose']
perc_cols = ['int_rate', 'revol_util']
date_cols = ['issue_d', 'earliest_cr_line', 'last_pymnt_d']
# Ensure that we have types for every column
assert set(cols_to_pick) - set(float_cols) - set(cat_cols) - set(perc_cols) - set(date_cols) == set(["id"])
# All categorical columns other than "loan_status" will be used as discrete features
discrete_features = list(set(cat_cols) - set(["loan_status"]))
# All numeric columns will be used as continuous features
continuous_features = list(float_cols + perc_cols)
```

In [13]:
```
for i in float_cols:
    final_data[i] = final_data[i].astype(float)

def clean_perc(x):
    if pd.isnull(x):
        return np.nan
    else:
        return float(x.strip()[:-1])
for i in perc_cols:
    final_data[i] = final_data[i].apply( clean_perc )

def clean_date(x):
    if pd.isnull(x):
        return None
    else:
        return datetime.datetime.strptime( x, "%b-%Y").date()
for i in date_cols:
    final_data[i] = final_data[i].apply( clean_date )

for i in cat_cols:
    final_data.loc[final_data[i].isnull(), i] = None
```

Clean Data

Same day return and outliers

In [98]:
```
final_data['loan_length'] = (final_data.last_pymnt_d - final_data.issue_d) / np.timedelta64(1, 'M')
n_rows = len(final_data)
final_data = final_data[final_data.loan_length != 0]
print("Removed " + str(n_rows - len(final_data)) + " rows")
```

```
Removed 2553 rows
```

In [99]:
```
# There are quite a few outliers, but the two most obvious
# ones to remove are in annual_inc, revol_util Remove these.
n_rows = len(final_data)
final_data = final_data[final_data.annual_inc < 10999200]
final_data = final_data[final_data.revol_util < 300]
print("Removed " + str(n_rows - len(final_data)) + " rows")
```

```
Removed 263 rows
```

Clean Data

Recent Loans, null values

In [100]:
```
# Remove all loans that are too recent to have been paid off or
# defaulted
n_rows = len(final_data)
final_data = final_data[final_data.loan_status.isin(['Fully Paid','Charged Off','Default'])]
print("Removed " + str(n_rows - len(final_data)) + " rows")
```

```
Removed 195705 rows
```

In [101]:
```
# Deal with null values. We allow cateogrical variables to be null
# OTHER than grade, which is a particularly important categorical.
# All non-categorical variables must be non-null, and we drop
# rows that do not meet this requirement
required_cols = set(cols_to_pick) - set(cat_cols) - set(["id"])
required_cols.add("grade")

n_rows = len(final_data)
final_data.dropna(subset = required_cols ,inplace=True)
print("Removed " + str(n_rows - len(final_data)) + " rows")
```

```
Removed 547 rows
```

Returns

Early repay – pessimistic approach

In [103]:
```
# Calculate the return using a simple annualized profit margin
# Pessimistic definition (method 2)

final_data['term_num'] = final_data.term.str.extract('(\d+)',expand=False).astype(int)
final_data['ret_PESS'] = ( (final_data.total_pymnt - final_data.funded_amnt)
                                        / final_data.funded_amnt ) * (12 / final_data['term_num'])
```

Returns

Early repay – optimistic approach

In [104]:
```
# Assuming that if a loan gives a positive return, we can
# immediately find a similar loan to invest in; if the loan
# takes a loss, we use method 2 to compute the return

final_data['ret_OPT'] = ( (final_data.total_pymnt - final_data.funded_amnt)
                                        / final_data.funded_amnt ) * (12 / final_data['loan_length'])
final_data.loc[final_data.ret_OPT < 0,'ret_OPT'] = final_data.ret_PESS[final_data.ret_OPT < 0]
```

Exercise #2

- Returns - a realistic approach
- The leakage problem
  - last_pymnt_d
  - fico_range_high
  - fico_range_low
- Identifying updates to the data